PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04L 29/00

A2

(11) International Publication Number: **WO 00/19677**

(43) International Publication Date: 6 April 2000 (06.04.00)

(21) International Application Number: PCT/CA99/00876

(22) International Filing Date: 24 September 1999 (24.09.99)

(30) Priority Data:

60/101,857	25 September 1998 (25.09.98)	US
2,264,407	4 March 1999 (04.03.99)	CA

(71) Applicant *(for all designated States except US)*: WIRELESS SYSTEM TECHNOLOGIES, INC. [CA/CA]; Suite 601, 205 Richmond Street West, Toronto, Ontario M5V 1V3 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: SNELGROVE, William, Martin [CA/CA]; Apartment #603, 90 Sherbourne Street, Toronto, Ontario M5A 2R1 (CA). STUMM, Michael [CA/CA]; 3 Belvale Avenue, Toronto, Ontario M8X 2A6 (CA). DE SIMONE, Mauricio [CA/CA]; Apartment 702, 10 Queens Quay West, Toronto, Ontario M5J 2R9 (CA). TRUDEAU, Chris [CA/CA]; Suite 908, 424 Yonge Street, Toronto, Ontario M5N 2H3 (CA).

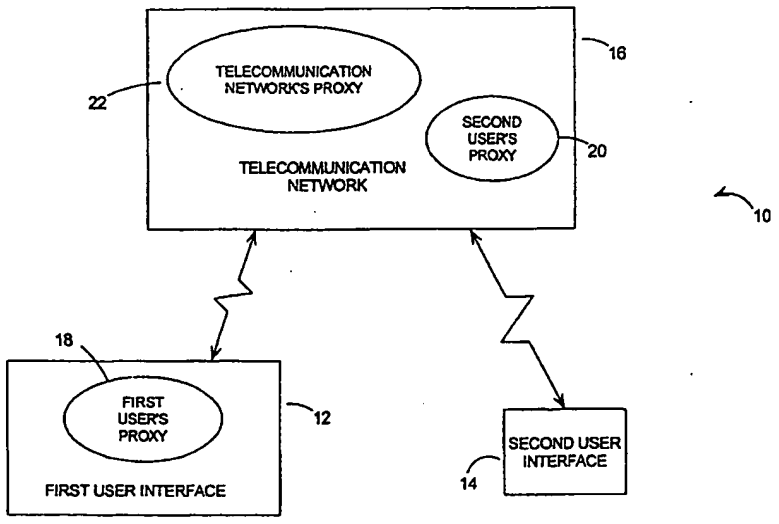
(74) Agents: O'NEILL, T., Gary et al.; Gowling, Strathy & Henderson, Suite 2600, 160 Elgin Street, Ottawa, Ontario K1P 1C3 (CA).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: CONNECTION MANAGER FOR TELECOMMUNICATIONS



(57) Abstract

The present invention relates generally to telecommunications, and more specifically, to a system and method of managing telecommunication connections. In telephony, the set up, maintenance and tear down of communications is implemented in a very inflexible and predetermined manner which impedes the addition of new features. The Internet is generally used as a simple transport network that is insensitive to performance requirements and therefore is a hindrance to high bandwidth services. The invention provides a system for managing connections which employs a proxy for each entity in the communication including each user and each telecommunication network. This allows a communication to be established with consideration for the resources available to each entity, in their respective limitations. This allows new services to be implemented over multiple networks, with the assurance that the necessary bandwidth and other resources will be available.

Connection Manager for Telecommunications

Field of Invention

The present invention relates generally to telecommunications, and more specifically, to a system and method of managing telecommunication connections.

Background of the Invention

The present invention relates generally to telecommunications and telecommunications systems which includes networks such as the public switched telephone network, Internet, cellular telephone systems, satellite communications, local area networks (LANs) and wide area networks (WANs). The interconnections which make up these networks may consist of a variety of media including optical fibre, wireless or hardwired electrical connections. As well, communications over these networks may be executed using a variety of protocols.

One major design aspect of communications in any of these networks is connection management, often referred to as "call processing" in the telephony world. Connection management is the process of setting up connections, managing connections in progress, and tearing them down. In telephony, for example:

- setup involves: detecting dialled digits, determining where the telephone being called is located, determining whether the called telephone is in use, and ringing it;
- managing a call in progress involves: detecting "flash" signals and handling "call waiting" processing; and
- teardown involves: releasing communication trunks and forwarding the billing record to the party being charged for the service.

Many telecom features, such as call-waiting and busy-call return, are primarily issues of connection management.

Connection management exists in the Internet, too. Internet protocol (IP) is in principle "connectionless" as each data packet includes a destination address and may arrive at its destination via a different route than other data packets transmitted during the same session, between the same calling party and called party. However, most IP communications are performed using a "virtual connection", such as Transmission Control Protocol (TCP) which is often used over IP. While IP handles the actual delivery of the data, the TCP keeps track of the individual data packets,

creating a virtual connection which lasts the life of the communication session. Similarly, other Internet applications such as chatting and Internet games may apply a similar concept, using protocols such as Internet Relay Chat (IRC) which uses IP as a pipe.

Regardless of whether communication within a network is considered connection-less or connection-based, some manner of connection management must be performed in order to administer communications within the network. In fact, when a telephone call is made today, the voice signal may be turned into data and follow a "connectionless" route anyway.

Connection features are at present complex to develop. Connection features interact with each other, and this feature interaction makes development and deployment of features difficult. As a simple example, "call waiting" is inconsistent with "voice mail on busy" and with "busy call return" because these features all specify different behaviours when the called line is busy. Compromise behaviours can be arranged when the features are inconsistent, but these situations must be specifically considered.

Features interact in particularly undesirable ways when different parties to a call have different interests. "Call forwarding", for example, involves one party using another's telephone to receive incoming calls, and the party to whom the call is forwarded may wish to control this.

Because new connection features are difficult to develop, telecom systems are slow to react to market needs. New media and new applications of communications are making feature development yet more complex.

Feature development is already difficult for the simple application of conversation over voice telephones. As new media such as videophone, typed messaging, shared files and whiteboards, and new applications such as distance learning, Internet Relay Chat, networked document preparation, meeting management and Internet gaming, develop, the problem is becoming even more severe. This problem will grow even greater as media are mixed, for example when voice-text conversion permits a mixture of media in a conversation, and when expectations develop for features from one domain to be mapped into another, as when customers expect a feature similar to call-waiting to apply in videoconferencing or Internet gaming.

As well, even for a single application, different users may have different needs, for example, requiring different degrees or forms of encryption. This makes the development of communications applications slow due to the complexity of handling many cases.

Telecommunications systems, such as those for telephony and the Internet, are composed of terminal equipment such as telephones or personal computers, an access network such as a telephony local loop or a radio link, and a backbone network such as the public switched telephone network (PSTN) or the intercity data networks. Although the needs of users at the terminals are very varied, the backbone networks must handle highly standardized loads in order to operate reliably and efficiently.

Telecommunications systems need to process the data flowing through in complex ways, often with processing occurring on computer systems separated both geographically and administratively. Many communications paths are simultaneously active, and the processing applied to the various flows of data changes frequently and in a wide variety of ways. The software needed to control these computer systems is generally large, complex and difficult to change.

When the data flowing through the system represents voice, such as in a modern digital telephone network, special processing must be applied to implement such features as three-way or multi-way calling, voice-mail, voice recognition and authentication, call waiting, encryption, voice coding and dual-tone multi-frequency (DTMF) detection. For data applications in general, such as electronic mail, remote computing, file transfer between computers or Web browsing, there are needs for security functions such as firewalls and encryption as well as datastream functions such as traffic shaping, error handling, prioritization, caching, format translation and multicast.

In voice telephony, services are implemented by having large computer programs running on centralized switches which interrogate local and distributed databases. The local databases specify which features are enabled on a given line, the switch software interprets these feature lists and implements the switch behaviour, and the switch software also interrogates the distributed databases via Common Channel Signaling System No. 7 (SS7) queries. SS7 is a global standard for telecommunications that defines the procedures and protocol by which network elements in the public switched telephone network (PSTN) intercommunicate control

messages for basic call setup, management, and tear down, as well as for special intelligent or database services such as local number portability (LNP), toll-free (800/888) services, and enhanced call features such as call forwarding, calling party name/number display, and three-way calling.

In PSTN, the connection manager is fixed by the services provided by the local exchange carrier, which in turn may only function within the bounds of the SS7 protocol. Therefore, calls can not access the switches except in a limited way, and new features can not be added by outside parties.

Telephony features, such as call forwarding, may only be implemented by adding code to the programs running the switches or by adding specialized hardware to the telephony network. The features available to particular users are defined in the local databases accessed by the switch software, and adding a new type of feature may involve changing these databases together with all of the switch software that uses them, and may also involve purchasing and installing new types of hardware in the network. Specialized software is also used to check the consistency of the features assigned to a particular user. For example, call-waiting and call-forward-on-busy features define different behaviours for the same event, a busy receiver; so both features may not be assigned to a user simultaneously.

In general, signal processing for telephony is done in hardware specialized for each type of task, for example, there is different hardware for tone decoding and conferencing. This limits the speed with which new features can be introduced since new hardware has to be designed, tested, manufactured and deployed. The fixed assignment of tasks also makes it impossible to share loads between different types of hardware, for example to use idle tone-decoding hardware to help with an overload of voice-conferencing.

Changes to existing telecommunication networks are therefore very complicated to make. There is a rigid model and hardware structure is difficult to extend. Therefore, existing telcos can not offer new features such as high quality voice. As well, existing telco's take a long time to bring such features to market.

The complexity of present telecommunications systems software, and the extensive interactions between its software components, makes the development of new features very difficult. As well, telecomm services have traditionally been provided by large monopolies who employed proprietary equipment that only they

had access to. Another complexity is that new services had to be backward compatible to handle their existing clientel.

Software development is therefore limited to a "closed" group of trusted developers, which reduces the talent pool available and shuts out developers with new ideas for niche markets.

Traditional telecomm does not consider differentiation, but focuses on provision of single services. Therefore, telecomm providers would not be encouraged to offer varied services at a cost reduction to users, for example, reduced quality of voice telephony on Christmas Day, simply to provide additional connections or reduced cost. As well, small niche markets have gone unserved completely as the cost of developing and implementing the additional products does not net sufficient profits.

Users can exercise a small degree of control over their telecommunications by use of software running on their personal computers (PCs). For example, there is currently a Telephony Applications Programming Interface (TAPI) that allows software running on a general-purpose computer to control the switching decisions of a type of switch known as a private branch exchange (PBX).

An application programming interface (API) converts a series of comparatively simple and high level functions into the lower level instructions necessary to execute those functions, simplifying use of an operating system. Using Windows APIs, for example, a program can open windows, files, and message boxes, as well as perform more complicated tasks, by executing single instructions. Therefore, TAPI consists of a collection of specialized subroutine calls that allow a user to set up and tear down circuits connecting particular physical devices, including telephone sets and servers for functions such as voice-mail. It also allows the user to define how the system should respond to events such as hangups.

A system known as Parlay implements a telephony API that can be used to control the central office telephone switches owned by large telephone companies. This is similar in concept to the use of a telephony API to control a PBX, but security concerns are of prime concern because of the number of telephone users who would be inconvenienced by a failure. Switch owners will not allow user access that may effect the reliability of their networks, therefore the degree of control that a user may exercise over his communications, is very limited.

Parlay, TAPI and similar systems permit third parties a degree of control over how telephone switches interconnect end users and specialized equipment such as voice-conferencing servers, but do not allow third parties to add new features such as encryption or voice coding. They are also unable to describe the handling of Internet traffic, and so it is necessary for a distinct system to be used to handle such functions as routing Internet browsing data through computers acting as security firewalls.

Networks for telephony and data transmission have developed separately, but the economic rationale for having distinct physical networks is very weak and therefore the technologies are converging. They appear to be converging on a model closer to that for data than that for telephony, partly because of its greater generality. The dominant data network is currently the Internet.

The Internet is a connectionless network service, in that a single communication may be broken up into a multitude of data packets that follow different paths in flowing between the same source and destination, though as noted above, various protocols exist which create "virtual connections'. Traditional telephony in contrast, establishes a single path that all of the data in the communication follow.

Regardless, call management of Internet applications is generally done by the software running at the endpoints, and the IP network is treated merely as a conduit for transfer of the data packets between the two points. The routers in the IP network merely index internal routing tables using the address in the data packet so that they know how to forward it, and do not generate data for either of the endpoints, or react to instructions from either of the endpoints. The Internet itself may be envisioned as a series of routers interconnected by an Internet backbone network designed for high-speed transport of large amounts of data. Users may access the Internet using personal computers in a number of manners including modems connected to the Public Switched Telephone Network, and set top boxes connected to existing telephone or television cable networks.

Communications over the Internet can take the form of various protocols, over a variety of physical transfer media. A protocol is a set of conventions or rules that govern transfer of data between hardware devices. The simplest protocols define only a hardware configuration while more complex protocols define timing, data formats, error detection and correction techniques and software structures.

Socket mechanisms are widely used to describe connections between applications programs running on operating systems such as UNIX and Windows. It

can be used to set up connections between applications programs running on different computers, such that packets of data are passed between them across such networks as an Ethernet or the Internet. When using a socket to communicate with a process on another computer, the programmer defines one side of a communication but must rely on the administrators of the other computer to have set up the other side.

Sockets typically use the Internet Protocol (IP) and can further be set up to use either the Unreliable Datagram Protocol (UDP) which sends packets without checking to see if they have been received, or the Transport Control Protocol (TCP) which will retry until it receives a confirmation of receipt. Telephony applications typically use UDP, because data that does not arrive on time is of no value, while file transfer programs typically use TCP so that accurate delivery is assured. The user is generally required to choose between these two mechanisms to specify handling of error conditions in packet delivery. Just as for telephony, it is difficult to add encryption or signal processing features to the handling of an IP stream.

The key advantages of a protocol like IP are that it allows a large network to function efficiently and that it offers a standardized means by which applications software can use that network. Disadvantages are that it does not allow specification of processing to be performed on data streams and that it does not accurately specify requirements on quality of service.

The Internet generally will not let a user run an applet on a node or server. This limitation is due to the architecture of the Internet which is based on the international OSI (Open Systems Interconnection) standard. The OSI standard describes communication systems using a seven layer model, each layer being operable to perform certain functions. Although OSI is not always strictly adhered to in terms of keeping related functions together in a well-defined layer, most telecommunication products make an attempt to place themselves in relation to the OSI model. The OSI standard is not likely to change dramatically, nor is the Internet's use of the standard, so the Internet will not likely become an active component in the provision of telecommunication services.

However, some protocols such as RSVP do exist, which set tags and priorities which can influence the routers a little, but not much. The resource reservation protocol (RSVP) is an extension to IP that permits specification of quality of service at a technical level, in terms of parameters such as data rates and

latencies. It has had limited acceptance due to the complexity it adds to backbone networks and the need for their switching hardware to be updated.

Therefore, typical software applications operating over the Internet look at the Internet as simply a transport network without any processing capability, and all functionality is placed at the calling and called party locations. Implementations of voice over Internet, for example, have software at the user's computer that acts as the interface with the user, converting voice to data packets for IP transmission, and if desired, may generate dial tones and DTMF tones simply as feedback for the benefit of the user. These tones do not pass over the Internet as all Internet communications are simply data packets with the destination attached.

Asynchronous Transfer Mode (ATM) networks use standard protocols for addressing packets of data and setting up connections, as do IP and TCP respectively. ATM networks have typically been deployed in the core of backbone networks because of the high speeds at which ATM equipment operates, and their capabilities have not been directly visible to end users. Because ATM routers are not directly accessible and because of the complexity of their mechanisms for describing quality of service (QoS), these mechanisms have not been used by applications software.

Besides the IP and ATM networks mentioned above, there are other data networks such as Frame Relay and Ethernet. As well, the PSTN may also be used to carry data, for example using trellis coding which maps digital data onto an analogue signal. Variants are also evolving of each major type of network, and engineering differences between implementations of these networks result in different performance. The complexity induced by this variety makes it difficult for users and application software to exploit all the networks available, and to exploit any to its fullest extent.

As explained above, existing data or voice networks are complex and it is either difficult or impossible for users to add new functionality to them. The convergence of these forms of networks, and the demand for services to move seamlessly between networks will make the task of providing new services in a mixed network environment even more difficult.

The access networks known in the art have severe limitations that come from their having been designed for narrowly defined telecommunications applications, such as telephony or file transfer. Therefore, an invention which allows an access

network to have the sophisticated functionality necessary for a mixture of telecommunications services is required.

There is therefore a need for a method and system of connection management for telecommunication systems that is addresses the complexity of such networks and provides an open, scalable and flexible architecture for the provisions of new services. This design must be provided with consideration for the pervasiveness of the existing infrastructures.

Summary of the Invention

It is therefore an object of the invention to provide a system and method of managing telecommunication connections that improves upon the problems outlined above.

One aspect of the invention is broadly defined as a connection management system for telecommunications comprising: a first user terminal; a second user terminal; a telecommunications network interconnecting the first user terminal and the second user terminal; and modular connection management software including a connection management software proxy for each of the first user terminal, the second user terminal and the telecommunications network, each of the connection management software proxies being operable: to execute on the system; and to provide the functionality to represent its owner's interests in managing the telecommunications connection.

Another aspect of the invention is defined as a method of connection management for a telecommunications system, executing on a telecommunications server, comprising the steps of: interconnecting a first user terminal and a second user terminal; and executing modular connection management software including a connection management software proxy for each of the first user terminal, the second user terminal and the telecommunications server, each of the connection management software proxies being operable: to execute on the telecommunications system; and to provide the functionality to represent its owner's interests in managing a telecommunications connection.

Another aspect of the invention is defined as a telecommunications server for a telecommunications system comprising: interconnecting means for interconnecting a first user terminal and a second user terminal; and modular connection management means including a connection management software proxy for each of

the first user terminal, the second user terminal and the telecommunications server, each of the connection management software proxies being operable: to execute on the telecommunications system; and to provide the functionality to represent its owner's interests in managing a telecommunications connection.

An additional aspect of the invention is defined as a computer data signal embodied in a carrier wave, the computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of: interconnecting a first user terminal and a second user terminal; and executing modular connection management software including a connection management software proxy for each of the first user terminal, the second user terminal and the telecommunications server, each of the connection management software proxies being operable: to execute on the telecommunications system; and to provide the functionality to represent its owner's interests in managing a telecommunications connection.

A further aspect of the invention is defined as a computer readable storage medium storing a set of machine executable code, the set of machine executable code being executable by a computer server to perform the steps of: interconnecting a first user terminal and a second user terminal; and executing modular connection management software including a connection management software proxy for each of the first user terminal, the second user terminal and the telecommunications server, each of the connection management software proxies being operable: to execute on the telecommunications system; and to provide the functionality to represent its owner's interests in managing a telecommunications connection..

Brief Description of the Drawings

These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings in which:

Figure 1 presents a schematic diagram of an exemplary telecommunications system in a broad embodiment of the invention;

Figure 2 presents schematic diagram of an exemplary mixed-network telecommunications system in an embodiment of the invention;

Figure 3 presents schematic diagram of a telecommunications system in the preferred embodiment of the invention;

Figure 4 presents a block diagram of a high level overview of an established connection in a preferred embodiment of the invention;

Figure 5 presents a block diagram of how a Connection Object establishes contact with a user in a preferred embodiment of the invention;

Figure 6 presents a block diagram of how a Connection Object is established in a preferred embodiment of the invention;

Figure 7 presents a block diagram of how a Connection Object establishes a negotiation in a preferred embodiment of the invention;

Figures 8A and **8B** present a flow chart of a process for connection management in a preferred embodiment of the invention;

Figure 9 presents a block diagram of a call-processing state machine responsible for initiating calls according to a conventional dial-tone model in an embodiment of the invention;

Figure 10 presents a block diagram of a call-processing state machine with detailed billing in an embodiment of the invention; and

Figure 11 presents a block diagram of a call-processing state machine with multiple independent bills in an embodiment of the invention.

Description of the Invention

A system which addresses the objects outlined above, is presented as a block diagram in **Figure 1**. This figure presents a block diagram of an exemplary connection management system **10** for telecommunications in which a connection may be established between a first user terminal **12** and a second user terminal **14** interconnected via a telecommunications network **16**. The phrase "terminal" is used generally in the art to describe any suitable manner of user interface in an audio, video, data or other similar fashion, and may include any type of telecommunication device including telephone, personal computer, cellular telephone, pager, fax machine of other device as known in the art.

The connection itself is managed by modular connection management software which includes a connection management software proxy for each entity in the system. In this case there are three entities with three corresponding proxies. The first user's proxy **18** is shown to reside on the first user terminal **12** and the second user's proxy **20** and telecommunications network proxy **22** are shown to reside on the telecommunications network **16** but as will be explained, they may

reside in other locations. Each of these connection management software proxies is operable to execute somewhere on the telecommunications system and to provide the functionality needed to represent its owner's interests in managing the telecommunications connection.

In this example, the first user's proxy **18** is shown to reside on the first user terminal **12** which requires first user terminal **12** to have the functionality to execute the first user's proxy **18**. The first user terminal **12** may, for example, comprise a microphone and speaker connected to the sound card of a personal computer, or a smart telephone with a built-in operating system. Respectively, the first user's proxy **18** could execute on the microprocessor in the personal computer or on the digital signal processor within the smart telephone.

The second user's proxy **20** is shown to reside on the telecommunications network **16** which may be necessary if the second user terminal **14** does not have the functionality to support the software proxy. The second user terminal **14** could, for example, be a standard telephone or cellular telephone. Nonetheless, the second user is still given symmetric treatment along with the other entities in the system, by having a proxy provided for it which executes on the telecommunications network **16**. In such a case, the second user's proxy **20** would generally be selected from a schedule of "generic proxies" by the telecommunications network **16** or the first user's proxy **18**. Telecommunications networks **16** that have the functionality to store and execute proxies are generally described herein as "active networks".

In fact, all users outside the system need "generic proxies" representing legitimate uses of the PSTN, for example, or to represent a PSTN calling party's requirements to a connection management proxy, such as caller ID. As an advanced feature, users may also be able to specify a special proxy to be used by an outside user.

A user with a direct connection to the active network who "roams" into the PSTN will also want to be able to inform his/her proxy of how he/she can be found. This could be done by the user accessing the active network via the PSTN, then authenticating to the active network, and finally by the user providing an Internet domain name to identify the Service Broker with which the User Proxy is associated so that the proxy can be located.

This connection management architecture is modular in that each entity is represented by a separate and independent software proxy which represents its

interests. If more entities are required to establish a connection, for example, to add participants to a conference call, or to use additional signal carriers, additional proxies are added to represent each participant. As will be explained hereinafter, this modularity simplifies the complexity of the system, which allows advances such as openness to new functionality. This modularity also allows for a separation of concerns, in that each entity need only be concerned about his own requirements. Again, this reduces the complexity of the system.

Proxies define a user's policies for creating and receiving calls, which deal with, for example:

- incoming calls, deciding which calls may ring a telephone, which should be returned busy signals and which go to voice-mail;
- billing, representing the end user's policies on what to pay for, such as 900 calls and disk space for voice-mail and how to pay for those services, such as requesting a bill or having the costs charged to a credit card account; and
- outgoing calls, using special "speed-dial" directories or voice-dialling software.

Generally, any party that may pay bills or charge for services may have a proxy. The invention is generally described with respect to a model of one proxy per owner, but other ownership models could also be used. Proxies can also represent groups of users such as local exchange carrier or long distance toll operators, and 911 dispatch operators.

The strength in the invention's generality becomes clear from the application to a mixed protocol network as presented in **Figure 2**. This figure presents a conference call between three parties **24**, **26** and **28**, in which two of the parties **24** and **26** have direct access to the active network **30**, and the third party **28** is connected to the PSTN **32**, accessing the active network **30** via an Internet backbone **34**. The two parties **24**, **26** with access to the active network **30** are connected to personal computers **36**, **38** which are located physically close to their telephones, for example, in the same house or office. The personal computers **36**, **38** are shown to be on the active network **30** because they are supporting and executing their corresponding proxies.

The active network **30** is also shown to include four nodes **40**, **42**, **44**, **46** but any number may exist, and they may be interconnected in any manner. The proxy for the active network **30** is shown to reside on node **42**, but could reside on any of the four nodes **40**, **42**, **44**, **46**. It is desirable to balance the loading of the nodes by

spreading proxies around, so proxies may be instantiated with respect for the loading on the nodes at the time of instantiation, and may also be relocated if a particular node becomes overloaded at any time during a connection. Such multitasking and load management techniques are known in the art.

The proxies for the Internet backbone **34**, PSTN **32** and third user **28**, are shown to reside on node **46**, but again, they may reside on any of the four nodes **40**, **42**, **44**, **46** on the active network **30**. From the perspective of the invention, the most significant point is that proxies have been created for entities which are not on the active network **30**, and that would not otherwise have the functionality to perform the additional processes. When instantiating proxies, it is desirable to locate agents as close as possible to the corresponding entity, to minimize the time required for intercommunication, but this must be balanced in consideration for whether this causes too great a load on a single node, compromising its real time operation. If a scheduler in a real time node recognizes that it will not be able to execute in real time, that is, it becomes overloaded, one or more proxies or their agents may be off loaded to another node, such as **40** or **44**. Again, such load balancing technique are known in the art.

Development and addition of new features to a telecommunication system in the face of growing complexity to those networks, requires the separation of concerns between the entities involved, that the invention provides. It must be possible to develop software for a new telecommunications application without being concerned about its effect on other applications. The use of a proxy for each entity in an open telecommunication system accomplishes this and allows for the addition of new features by outside parties.

Additional aspects of the invention will be described which provide further advantages, but in the broad implementation, the use of proxies allows users to control the features they have access to. New proxies may be written, edited or downloaded from the Internet. Knowledge of the proprietary PSTN switches and switch software is not necessary, as the proxies may be written in any form of code. As noted in the background above, outside parties can not access the PSTN to add new features, or modify existing ones.

Also as noted above, new applications on the PSTN generally must take into account all other switch functionality, and legacy hardware and software, so that they may be reliably employed. As well, new features often require upgrades to software

and hardware at every switch, and possibly every port that is to offer the service. This could require upgrades to thousands of ports over a large area! Because each new feature may require such a tremendous investment of time and resources, PSTN providers focus on providing commodity products rather than differentiated products, and their products take a long time to be brought to market.

In contrast, each proxy need only be capable of providing a certain narrow functionality, so they may be far less complex. As well, proxies may be written by any developer, anywhere in the world. A single developer may write a proxy and make it available over the Internet, being compensated for its use via known electronic commerce or milli-commerce techniques. In the preferred embodiment proxies are assemblies of agents, reducing the complexity of the software even further.

As a simple example, the invention handles the problem of conflicting features, such as how to respond to a busy tone: by going to voice mail or call waiting? As noted in the background, the existing PSTN must be set up one way or the other. In order to switch modes, the user must contact the LEC and make a request for the service change, which may require manual changes to switch software or hardware, may incur a service charge, and may take a long time to be effected. As well, either or both of the services may simply not be available on the switch that serves the user. In contrast, with the invention, the user merely specifies how the situation is to be handled within his proxy. He has complete control over his proxy and access to it, so he may change it as often as he requires, simply by changing a software selection switch on his software interface.

The problems identified in the Background with respect to the Internet are quite different from those of the PSTN. While the PSTN has centralized control, the Internet heretofore has positioned control at the ends of the connection. This limits connections to situations where both end users are capable of executing their own control, complementary to one another, and all other parties, including the network itself, are precluded. Also as noted in the background, Internet software applications are generally provided as commodity products as well, and do not have the modularity to add new features or functionality.

As described above, the invention allows the network or networks to participate in the connection management process. As well, the modularity of the invention allows for easy addition of new features.

In the preferred embodiment of the invention which is described hereinafter, additional features are described which provide numerous other advantages. In general, a Proxy is defined as a state machine having a code structure that makes for clarity and allows some sort of intuitive interface. The sequence of actions involved in taking a call can be described by a state machine, and the state machine can be edited as a graph, one which is in turn responsible for setting up switching graphs. Those features and their corresponding advantages are summarized as follows:

1. **Proxies Call Multiple Agents**

   Proxies may contain multiple agents, each agent dedicated to a particular task, and the proxy need only call the agents it needs to perform a certain task. This provides further modularity, reducing the size of the proxy and making it easier to understand and modify, and to run faster.

   The terms proxy and agent are sometimes used interchangeably in the art. For the purposes of this document, they are distinct: a proxy is built out of agents, each of which handles a special situation. Therefore, the proxy does not comprise an immense block of code with all conceivable functionality, but in its simplest form, is merely a supervisor which instantiates software agents as required, discarding them when their task has been completed.

2. **Concurrency**

   The proxy for a particular entity may be invoked multiple times, to handle several calls arriving at the same time. This requires that the operating system be multitasking, which is known in the art, and that the proxy instantiate multiple agents. As well, multiple Filter Runtime Engines (FREs) must be instantiated, with one per connection. FREs are the software environment where filters and agents defined by the graph data packet execute and are described in greater detail hereinafter.

3. **Standard Application Programming Interface (API)**

   As noted above, an API converts a series of comparatively simple and high level functions into the lower level instructions necessary to execute those functions, simplifying use of an operating system.

   The particulars of how an API for the invention is implemented are not critical, but it is desirable that a standard API be employed that expresses control, connection and negotiation processes, including payment.

4. **Real Time Operating System (RTOS)**

Voice telephony is a real time procedure, so if the system is to be applied to voice, a RTOS should be used as known in the art. Generally, RTOS's schedule threads and functions. RTOSs specify deadlines prior to which blocks of software code must be executed, and use schedulers to ensure that those blocks of code are executed on time, and in proper coordination with other blocks of code.

5. **Distributed Operating System**

A distributed operating system is one in which portions of the software can run on different nodes. In the case of a telecommunications system, distribution of software makes it easier to maintain real time operation as there are more options available to schedule timely execution. As well, distributed operation improves scalability and speed. The use of agents and proxies lends itself to the efficient use of a distributed system, in that agents and proxies may be assigned to run on different nodes of the system. Ideally, agents will be located close to where they are required, to minimize time delays in communicating with the entities they represent. Such a suitable distributed RTOS is described in the co-pending patent application under the Patent Cooperation Treaty, Serial No._____, titled Distributed Telco.

6. **One Agent per Node**

It is desirable that a software agent be provided which is associated with each node of the system through which the connection passes. Firstly, this provides a uniform node platform in that all nodes appear to be uniform. Secondly, this allows for control and configuration of the nodes, for example, allowing them to raise exceptions if frame error rate is exceeded, latency is too long, or a similar communication requirement has been violated.

7. **Administrative Control**

It is preferred that an administrator have access to proxies and authority over them, allowing for maintenance and control of the system. This would include such functions as governing who can create proxies and where they may execute. Because the invention allows such pervasive functionality over all manner of telecommunication networks, there is a need for a party which can offer some measure of control. The Administrator's access would of course have to be very secure to prevent unauthorized tampering.

8. **Market Model for Agents**

It is preferred to encapsulated call processing within a market model. That is, having a separate software module corresponding to each of a user's voice, telephone, ISP, Qwest IP backbone, and other distinct products.

The market aspect refers to ownership of a resource and authority to control it; therefore, the proxies must identify each resource and the party which controls it. For example, an Internet Service Provider (ISP) may own switches and nodes, so his proxy administers those resources, while the user owns his own telephones so will not accept behaviour that infringes his right. The ISP will set an information rate which the user has no choice but to recognize. One can however, modify or delegate authority for one's own resources.

9. **Security Mechanism**

A strong security mechanism is required that protects proxies from erroneous or malicious code in other proxies, as well, pieces of proxies may be distributed to other users and the proxy owner may not wish for his proxy to be available to others. These security mechanisms could include digital signatures such as RSA, or other encryption and authentication techniques known in the art.

10. **Graph Model**

It is preferred that the invention be applied to a network which employs a graph model as described in the co-pending patent application under the Patent Cooperation Treaty, Serial No._____, titled "Method and System for Configuring Communication Resources". Briefly, the graph model describes a call as a small data structure that simply contains calls to desired filters and their locations, rather than containing all the code for the filters themselves. The graph data packet may be passed around the network, being reviewed and amended by various entities. When all or part of the graph is to be executed, a device simply assembles the listed filters in the manner defined in the graph data packet.

In addition to filters, it is also preferred that this graph data packet contain calls to proxy agents required to set up the call. Two examples of such an implementation are described herein with respect to **Figures 10** and **11**.

11. **Graphic User Interface (GUI)**

A GUI is piece of software that presents data to users in a graphical manner, allowing for easy interpretation and modification. It is preferred that the invention be implemented in such a manner, where possible. The GUI runs as Java in the PC browser, and communicates with call processing applications running on the active network by means of sockets. Invoking it involves typing a URL (uniform resource locator such as "coolPhones.com"), after which it sits in a window waiting for an incoming call or a user input event to place a call. Inputs can be made via a mouse, keyboard, trackball, touchscreen, joystick or other similar manner.

A GUI is particularly well suited to the use of a graph model as described above, as the GUI may present the assembled filters as defined by a graph data packet, to the user in a very logical and understandable form. It is also preferred that the GUI have the functionality to let the user modify the graph data packet simply by altering filters and their interconnections.

12. **Persistence**

Proxies should persist in the presence of component failures, so that, for example, a user's forwarding instructions do not get lost during a crash. It is preferred that persistence be provided via a distributed database which is continuously updated, so that all concerned parties are aware of the status of the communication. In the event of a failure, the system is able to work around the failure, allowing the communication to continue. Such transactional interaction techniques are known in the art.

13. **Java™ Code**

It is presently desirable that proxies and agents be written in Java™, but another language with similar advantages could also be used. Advantages of Java™ include:

i. excellent security
ii. a large community of experienced developers
iii. object oriented code structure
iv. simple net-based distribution mechanism

14. **Directories**

Directories can be implemented as distributed databases, for scalability, and custom versions can also exist. A directory really belongs to a user or group

of users, rather than to the network provider. In an open system, it is desirable for a private "yellow pages" to prosper, for example, and anyway it can not be prevented because it could be provided from any IP address, though less efficiently.

15. **Call Manager**

A "call manager" will be needed for setting up complicated types of calls. For example, one complication in a three-way calling is what to do when the originating party hangs up: does the call drop? If not, either the calling party may find itself handling a number of calls of which it is not part or it will delegate the task to a call manager. The call manager mechanism can also be used to simplify the task of call setup for simple calls.

16. **Suitability for Negotiation**

It is preferred that the architecture for agents provide for use of the negotiation system described in the co-pending Canadian Patent Application No. _____, titled "Method and System for Negotiating Telecommunication Resources".

A telecommunications system implemented with the functionality described above provides a foundation for the mixed media applications of the future, and for greater flexibility and power to existing services such as high bandwidth telephone, and Internet gaming.

The preferred embodiment is to apply the invention to a wireless access system as presented in **Figure 3**. In this figure, the telecommunications network **16** includes both a public switched telephone network (PSTN) **32** and Internet **48**. Part of the Internet network **48** is shown to comprise an asynchronous transfer mode (ATM) network **50**, but other telecommunication networks are also compatible with the invention including synchronous transfer, integrated services digital network (ISDN), asynchronous digital subscriber line (ADSL), local area networks (LANs), and wide area networks (WANs). Users may connect to this network **16** by use of hard wired telephones **52** or wireless telephones **54** which connect to the network **16** through base stations **56** of service provider BS-A. Note that these base stations **56** are interconnected via a backbone which may comprise hard wired interconnections, IP or ATM routers as shown, or other similar means.

Similarly, computers **58** could access either by wireless, through a base station **56** as shown, or be hard wired to the network **16**. Wireless connections of

telephones and computers to the network **16** are by means of patchpoints (PP) **60**. Redundant illumination, as shown with respect to computer **58** is preferred where possible. Also, telephones may have both wireless and hard wire access as shown with respect to location **62**.

These arrangements are shown simply as examples, and it would be clear to one skilled in the art that many alternative arrangements are also possible.

The software layer of the invention is independent of the hardware layer, allowing filters, proxies, agents and other software components to be located anywhere accessible in the system, dynamically mapped onto appropriate hardware, and executed.

In the preferred embodiment, a user proxy is software as described above, which represents the user's interests on the network. This process is described in greater detail with respect to **Figures 4 - 7**, but as an overview may be described as follows:

- That a user proxy is divided into several parts: a User Proxy object, multiple User Reactor objects, and multiple Connection Agents. The User Proxy object lives on the active network and is responsible for creating and coordinating the User Reactors and Connection Agents. The User Reactor is a piece of a proxy that reacts to an incoming event a user has an interest in and is responsible for handling the instantiation of a connection. The Connection Agent is responsible for ongoing connections.
- Methods are provided for User Reactor objects to evolve into Connection Agent objects when a newly instantiated connection becomes active. A single User Proxy can have multiple instances of User Reactors and Connection Agents each of which represent the user in a connection. A Connection Object is an object which corresponds to an active connection on the network and lives on the Service Broker. It is used to interface the various User Proxies and starts the negotiation process.
- A Service Broker is a software abstraction of where a connection and the connection components interact. Therefore, a Services Broker houses the User Proxies and any active Connection Objects.

As well, the implementation as described herein is based on the following assumptions:

- All User Proxies are instantiated at start-up and live on a single server of the active network
- All software components are well behaved
- All software components are trusted

Figure 4 presents a high level overview of an established connection in an embodiment of the invention. Each of the two users has an associated Filter Runtime Engine (FRE) **64** and a Controlling Application **66** running on its terminal. Such a pairing is referred to as an Aware Application **67**. Each of the two users also has an associated User Proxy **68** which represents its interests and resides on the active network. These two User Proxies **68** intercommunicate with one another via the Connection Object **70**.

A Controlling Application's **66** contact with a User Proxy **68** is always established through a User Reactor **72** as shown in **Figure 5**. A Controlling Application **66** is one part of the software which controls a terminal on the active network. It sits on top of the Filter Runtime Engine (FRE) **64** and acts as an interface between the filters and the User Proxy **68**. The Filter Runtime Engine (FRE) **64** is the software environment where filters and agents defined by the graph data packet execute. The User Reactor **72** is instantiated when the Controlling Application **66** sends an event to the Service Broker **74** as noted above.

As filters and agents execute on the FRE **64**, they may pass events to the Controlling Application **66**, which in turn, passes methods and events back to the FRE **64**. In other languages methods are called functions, subroutines or procedures. A Java™ method make take any number of parameters or none at all, and may or may not produce a result, but may never return more than one result. All methods in Java are part of some class, and there are no stand-alone methods, not even main.

Events are a more complex Java™ tool, and may be described as:

- a way one object can call a method in another, but with a delay. Instead of doing the work right away, the work is postponed by putting into a queue, so that the caller can get on with more important work;
- a flexible way of deciding at run time precisely which objects should have their methods called when something interesting happens; or

- a way for letting the callee decide when it wants to be called rather than leaving that decision totally up to the caller.

Events are transferred between components using a Channel. There are several implementations of Channels which allow local and remote transfer of events. Each of the components can be considered a layer in the event model and can pass an event to either the layer above or bellow itself. The layers, in order of typical event transfer, are as follows:

- FRE filter events
- Controlling Application
- User Proxy/Reactor/Connection Agent
- Connection Object
- Other participant's Connection Agents

User Reactors **72** are specific to a system and are typically used for two purposes: configuring a User Proxy **68**, and establishing a connection. When the user wishes to establish a connection the User Reactor **72** sets up the connection and then evolves into a Connection Agent **76** which is used for the duration of the call. The following is a list of the steps used to establish a connection.

1. Some Controlling Application **66** contacts its User Proxy **68** by opening a Channel **78** to it. A Channel **78** is a conduit for events used to communicate between various components in the system. The Channel **78** locates the user proxy using a unique identifier to identify the Service Broker with which the User Proxy **68** is associated. In the preferred embodiment a "distinguished name" is used, in accordance with the LDAP (Lightweight Directory Access Protocol) standard. The User Proxy **68** creates a User Reactor **72** and passes it the Channel **78**.
2. Next, the Controlling Application sends a message to the User Reactor **72** via the Channel **78**, which identifies the other participants in this connection. This identification may be their DN or some other mneumonic such as a telephone number, which it translates to a DN.
3. The User Reactor **72** then uses the Service Broker **74** to create a new Connection Object **70** as shown in **Figure 6**. The Service Broker **74** uses the Connection Factory to create the Connection Object **70**. A Connection Factory is the interface on the active network used to create Connection Objects **70**.

The creation of the Connection Object **70** includes a parameter which is a reference to the User Proxy **68**. The User Reactor **72** then adds the proxy for the other participant to the Connection Object **70** using their DN.

4. The Connection Object **70** contacts the User Proxy **68** of each of the connection's participants and requests a Connection Agent **76**. A different method is used for this request if the participant is the caller. This allows the User Proxy **68** to evolve the User Reactor **72** into a Connection Agent **76**. The Connection Agent **76** and Connection Object **70** are now able to exchange events and messages.

5. The Connection Object **70** creates a Floor Object **80** as shown in **Figure 7**, which takes a list of the participant's Connection Agents **76**. A Floor is an object which handles negotiation, which is preferably implemented as a five step process:

   a. **Information gathering**: each participant is allowed to declare information about the connection. This information is stored in name value pairs. The Floor also fills in certain information about the callers, like their DN, so they can not falsely represent themselves.

   b. **Terminal Determination**: each participant is asked to determine what terminal it will use in the connection.

   c. **Terminal Informing**: each participant is informed as to what terminals the other participants are using.

   d. **Negotiator Determination**: each participant is asked to for a Negotiation class. This is typically based on what Terminal the participant is using.

   e. **Negotiation**: the Negotiators **82** are put together into the Discipline **84** (determined in the Connection Factory) which results in PDL (Path Development Language).

   A more detailed description of the negotiation process is available in the co-pending Canadian Patent Application No. _____, titled "Method and System for Negotiating Telecommunication Resources".

6. The Connection Agents **76** are then responsible for taking the resulting PDL and sending it to their corresponding Controlling Applications.

As an example, **Figures 8A** and **8B** present a flow chart of a process for establishing a POTS (plain old telephone system) connection. At step **86**, system

start up is executed, which includes the start up of a Service Broker **74**, which instantiates an object for every User Proxy **68** for which it is responsible, and the start up of each user terminal.

At step **88**, terminal registration begins. The terminal registration process has been circumnavigated by a combination of command line parameters to the Aware Application **67** and information stored in the LDAP (Lightweight Directory Access Protocol) database. LDAP is an Internet protocol that email programs use to look up contact information from a server. Association with a specific User Proxy is set by hard coded information in the User Proxy's constructor and command line parameters for the application. The initial PDL is read out of a file.

Until the user's telephone or equivalent interface goes off-hook at step **90**, control will remain in a wait state by looping back to itself. When it does go off-hook, control passes to step **92**, where a filter reacts by generating an off-hook event. THe filter waiting for the off-hook could either monitor the telephone continuously, or wait for off-hook until it is instantiated.

This off-hook event may initiate the controlling application at step **94**, or it may already be installed on the system in which case it changes the state of the controlling application. Part of this state change may by the return of a dial tone to the user's telephone.

The user then begins to enter digits or other keystrokes which are recognized by a DTMF filter on the FRE as an incoming events at step **96**. The DTMF filter detects these digits and generates digit events which are collected by the Controlling Application **66**. The FRE sends events by raising an exception of DTMF code + #, and Java™ has a routine to catch exceptions, which acts something like a software interrupt.

At step **98**, the Controlling Application **66** considers whether it has enough digits to identify the called party, and if not, returns control to step **96**. If sufficient digits have been received, a Channel **78** is opened to the User Proxy **68** of the Controlling Application **66** at step **100**, which connects it to a User Reactor **72** and sends a start call message. The start call message includes the digits dialled.

At step **102**, the User Reactor **72** receives the start call message and translates the digits dialled string into a domain name (DN) for the called party. The User Reactor **72** then establishes a connection between the parties as described above.

Code Architecture:

The current architecture of the code is separated into three concerns:

- the core system software, which typically would only be altered by system administrators or close affiliates. This code includes the Service Broker, Connection Object, Floor, Channels, and PDL;
- the Software Development Kit which is a set of base classes and interfaces for use by developers. This includes abstract classes which define the concepts of User Proxies, User Reactors, Connection Agents, Negotiators and Disciplines. There are also some basic implementations of these abstract classes which make development easier. For example: the LookupCA is an abstract Connection Agent which has default actions for everything except the terminal determination. This allows a developer to extend the LookupCA, implement just the terminal determination method and have a working Connection Agent; and
- the remaining pieces of code required to implementation a complete, although basic, system.

Examples:

Voice-mail and its video and text equivalents are usually thought of as pure data, but should be seen as objects with methods for reading and writing, or as filters that persist after calls complete. The reason to generalize is to allow different types of coders including encryption and fax data, to be used in a flexible manner with voice-mail.

Therefore reading voice-mail can be thought of as accepting a call, albeit a time-shifted one. Voice-mails are all pending requests to the called party's proxy, and display the graph of the call that set them up, even though it has long since been torn down, so that the accepting party can see data type, coding and encryption needs, source of call, identify who is paying, and other such information.

A single-use proxy stays alive and attached to the mail, into which one may also call to retrieve mail. This permits group messaging or retrieval by password, situations in which the voice-mail does not know who to contact. In this sense reading voice-mail can be thought of as originating a call.

Figure 9 for example, presents a block diagram of a call-processing state machine responsible for initiating calls according to a conventional dial-tone model. In particular, this figure shows the filters that a user assembles to implement a

standard dialling model. This representation can be edited graphically, and different blocks substituted to allow voice dialling or different interpretations of numbers, for example, the PBX convention of dialling 9 to access an outside line or something where press-and-hold of a digit gives speed dialling.

A telephone device normally resides in a wait state **106**. If an incoming call arrives, the state shifts to process the incoming call **108**. If the telephone goes off hook, processing shifts to the dialling interpretation routine **110** which offers a dial tone to the user until the first keystroke is made, which causes a corresponding DTMF signal to be returned, and a change made to state **112**. Processing then awaits the entry of additional keystrokes which are interpreted in state **114** and corresponding DTMF signals are returned. Once the dialled number is complete, control passes to the call set up routine at state **116**. Call setup is executed as described above: set up your half of the call per steps **106 - 114**, then a call manager will make the connection to the called party or parties. The called party or parties may return withe counter proposals as part of the negotiation process, and the call manager adjusts the call graph as required, or allowed by the calling party's proxy. The user may have changes to he call graph flagged to him, or allow them to be accepted without review.

Figure 10 presents a block diagram of a call-processing state machine with detailed billing in an embodiment of the invention. Filters for billing services are simply interposed between the body of the call **118** and the ultimate source of money, represented in this case by a credit card company **120**. The call manager passes the billing service **122** a copy of the entire call graph on call setup so that it has all the necessary information to describe the call, and it also receives copies of warranty-failure alerts from the warranty monitoring filter **124**. It stores its results on rented disk, which can then be read through a Web interface or printed and mailed.

The call graph presented in this figure is between the source **126** and destination 1 **128**, and has been negotiated through a channel **130** which has guaranteed a certain minimum latency. As part of the call graph, the source **126** has therefore inserted a latency cop **132** which monitors the latency through the channel **130** and advises the warranty filter **124** if the channel **130** has failed to meet its obligations by sending it failure signals. The duration of the call is also sent by the timestamp **134** through the graph for use by the billing service **122**.

If there is a failure, the warranty filter **124** will affect some form of compensation to the source **126** and/or destination 1 **128**, which may, for example, be in the form of free transmission frames. This information is passed on to the billing service **122** which compiles the bill, forwarding it to the credit card filter **120**, which confirms the payment back through the other filters in the chain.

Similarly, **Figure 11** presents a block diagram of a call-processing state machine with multiple independent bills. Multiple billing sources are handled by attaching different money sources to different subgraphs. In this example, a broadcast-type transmission is being made from a source **136**, through a splitter **138** to two receivers: destination 1 **140** and destination 2 **142**. Each of the three parties has an independent credit card filter **144**, **146** and **148**, to settle the respective costs to each of the three parties. This model is useful in the case of point-to-multipoint services, like MBone or other pseudo-broadcasts.

Examples have been shown to demonstrate various aspects of the invention, but the number of variations is by no means complete. Comparable implementations could be made for any telephony device, including personal digital assistants, fax machines, pagers, point of sale computers, amateur radios, local area networks or private branch exchanges. While particular embodiments of the present invention have been shown and described, it is clear that changes and modifications may be made to such embodiments without departing from the true scope and spirit of the invention.

The method steps of the invention may be embodied in sets of executable machine code stored in a variety of formats such as object code or source code. Such code is described generically herein as programming code, or a computer program for simplification. Clearly, the executable machine code may be integrated with the code of other programs, implemented as subroutines, by external program calls or by other techniques as known in the art.

The embodiments of the invention may be executed by a computer processor or similar device programmed in the manner of method steps, or may be executed by an electronic system which is provided with means for executing these steps. Similarly, an electronic memory means such computer diskettes, CD-Roms, Random Access Memory (RAM), Read Only Memory (ROM) or similar computer software storage media known in the art, may be programmed to execute such method steps.

As well, electronic signals representing these method steps may also be transmitted via a communication network.

It would also be clear to one skilled in the art that this invention need not be limited to the described scope of computers and computer systems. The principles of the invention could be applied to citizen's band radio, amateur radio, or packet radio. Again, such implementations would be clear to one skilled in the art, and do not take away from the invention.

WHAT IS CLAIMED IS:

1. A connection management system for telecommunications comprising:

a first user terminal;

a second user terminal;

a telecommunications network interconnecting said first user terminal and said second user terminal; and

modular connection management software including a connection management software proxy for each of said first user terminal, said second user terminal and said telecommunications network, each of said connection management software proxies being operable:

to execute on said system; and

to provide the functionality to represent its owner's interests in managing the telecommunications connection.

2. A system as claimed in claim 1, wherein each said proxy is operable to provide the functionality to represent its owner's interests in managing set up, maintenance and tear down of the telecommunications connection.

3. A system as claimed in claim 2, wherein each said proxy is operable to provide the functionality to represent its owner's interests with respect to cost and quality of service in managing set up, maintenance and tear down of the telecommunications connection.

4. A system as claimed in claim 3, wherein each said proxy comprises:

multiple software agents each being operable to perform a specific task; and

a proxy object operable to instantiate particular ones of said multiple software agents in response to requirements of communications made over said telecommunications system.

5. A system as claimed in claim 4, wherein said telecommunications network comprises multiple telecommunications networks with varied protocols, each of said multiple telecommunications networks having its own connection management software proxy.

6. A system as claimed in claim 5, wherein said proxy object is operable to locate said multiple software agents for execution on devices closest to where they are required.

7. A system as claimed in claim 6, which is further capable of handling multiple communications for a given entity, comprising:
a multitasking operating system executing on said system; and
said connection management software proxy for said given entity being operable to instantiate software agents require for each of said multiple communications.

8. A system as claimed in claim 7, wherein each said operating system comprises:
a real-time multitasking operating system including a scheduler to administer timely execution of software threads and functions.

9. A system as claimed in claim 8, wherein each said operating system comprises an operating system having a standard applications programming interface (API).

10. A method of connection management for a telecommunications system, executing on a telecommunications server, comprising the steps of:
interconnecting a first user terminal and a second user terminal; and
executing modular connection management software including a connection management software proxy for each of said first user terminal, said second user terminal and said telecommunications server, each of said connection management software proxies being operable:
to execute on said telecommunications system; and
to provide the functionality to represent its owner's interests in managing a telecommunications connection.

11. A telecommunications server for a telecommunications system comprising:
interconnecting means for interconnecting a first user terminal and a second user terminal; and

modular connection management means including a connection management software proxy for each of said first user terminal, said second user terminal and said telecommunications server, each of said connection management software proxies being operable:

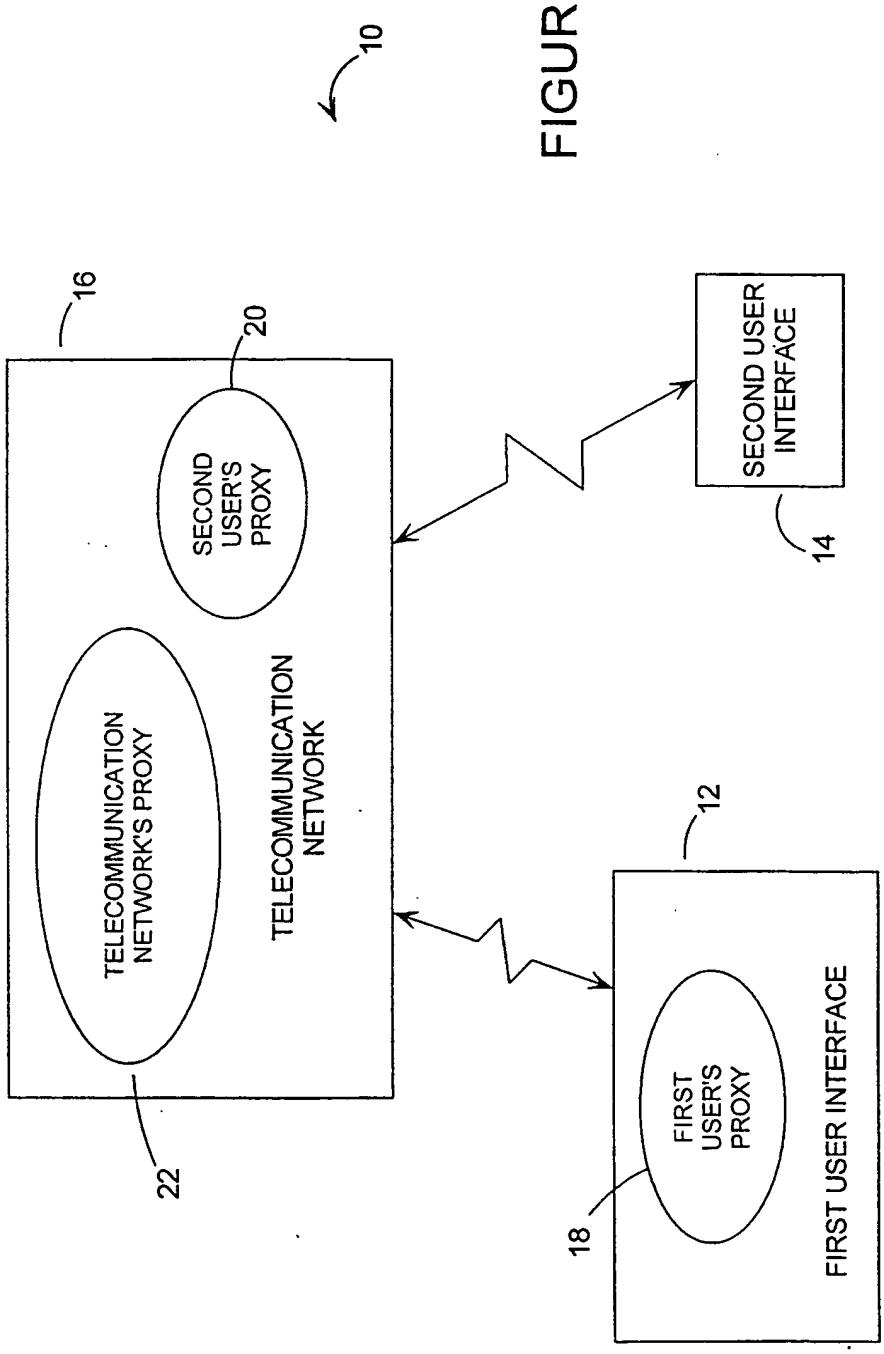
to execute on said telecommunications system; and

to provide the functionality to represent its owner's interests in managing a telecommunications connection.

12. A computer data signal embodied in a carrier wave, said computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of claim 10.

13. A computer readable storage medium storing a set of machine executable code, said set of machine executable code being executable by a computer server to perform the steps of claim 10.

FIGURE 1


10
16
20
SECOND USER'S PROXY
TELECOMMUNICATION NETWORK'S PROXY
TELECOMMUNICATION NETWORK
22
SECOND USER INTERFACE
14
12
FIRST USER'S PROXY
18
FIRST USER INTERFACE

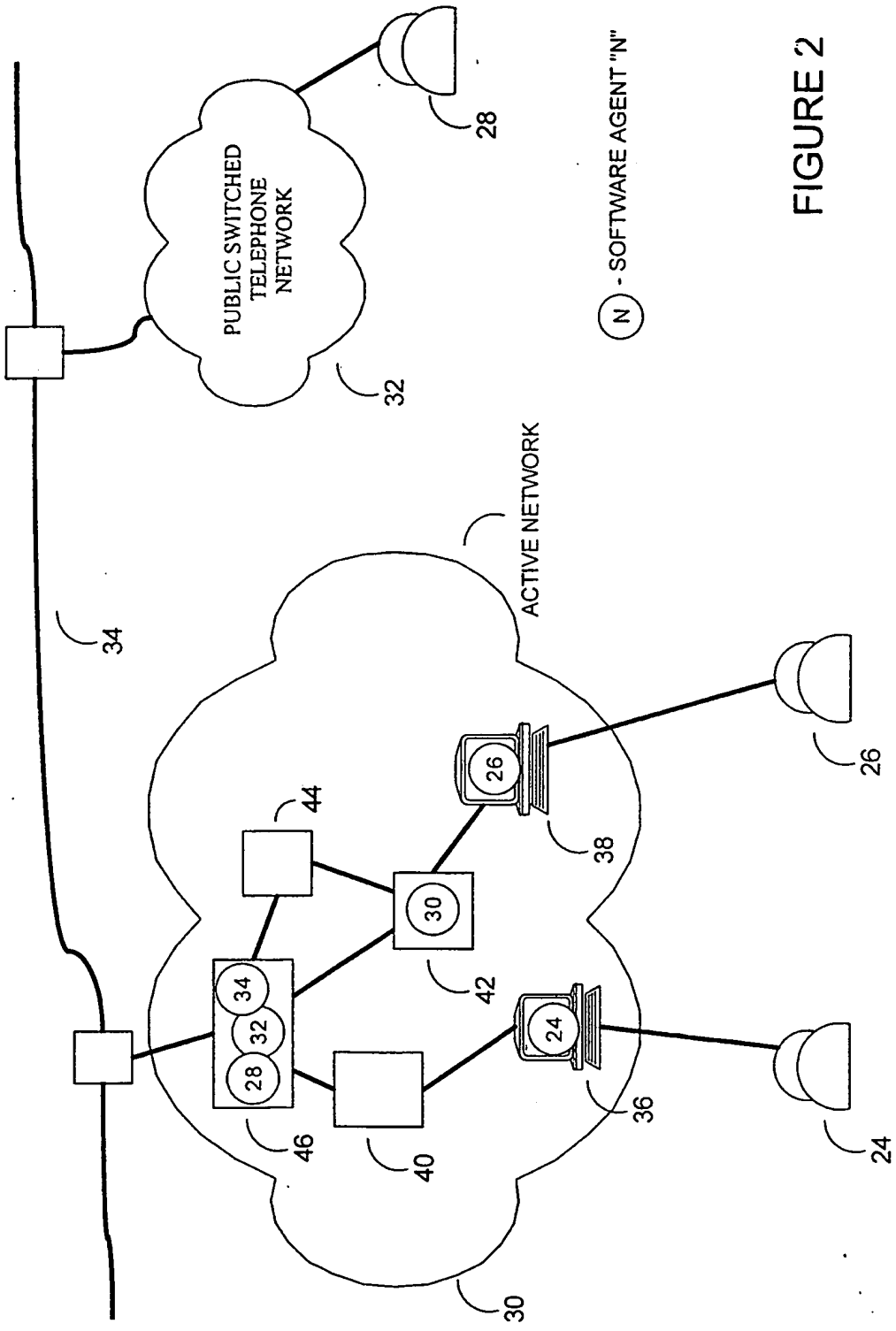

PUBLIC SWITCHED TELEPHONE NETWORK
28
32
34
ACTIVE NETWORK
(N) - SOFTWARE AGENT "N"
26
38
44
30
42
34
32
28
46
40
24
36
30

FIGURE 2

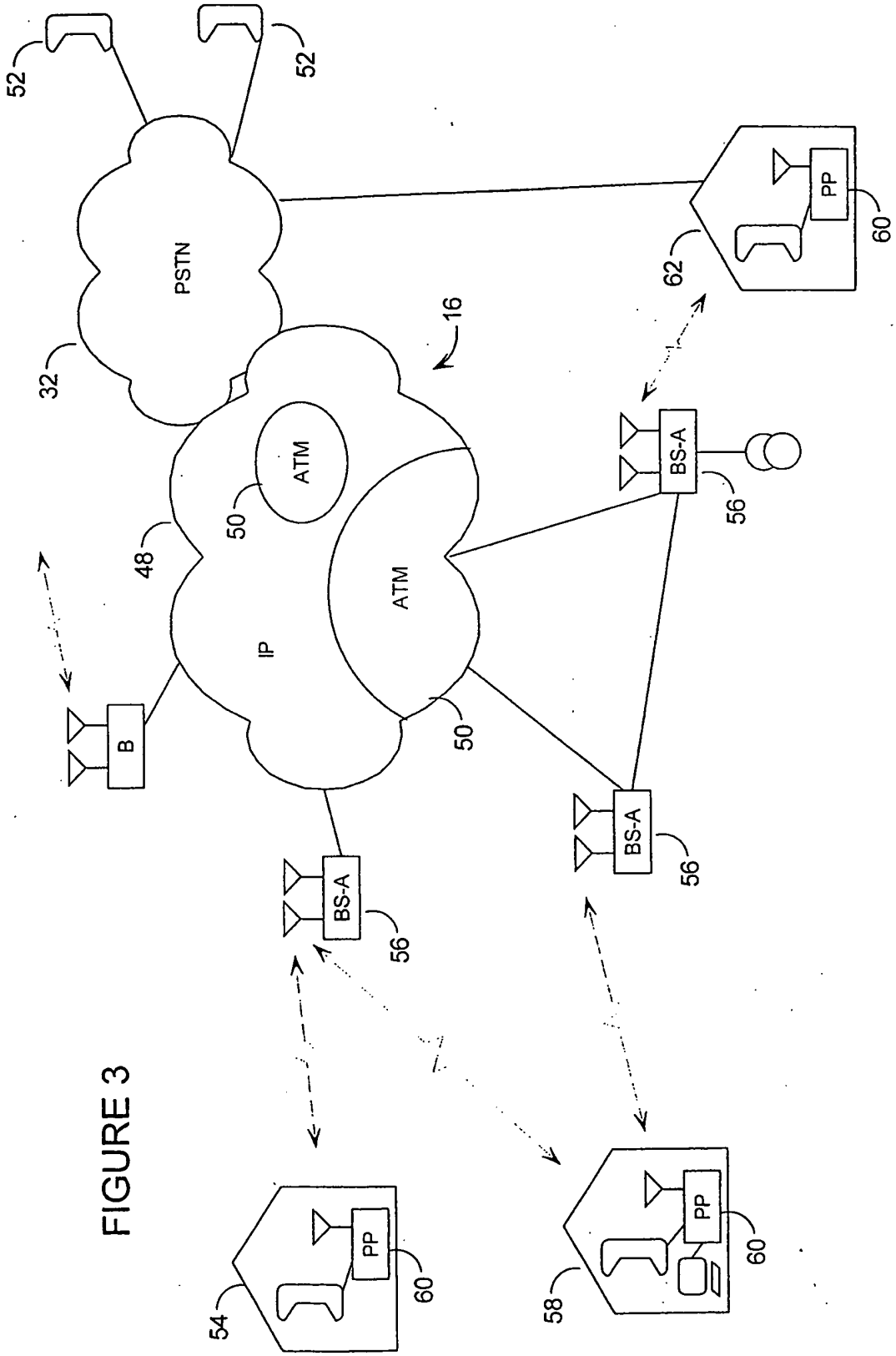

52
52
PSTN
32
16
ATM
50
48
IP
ATM
50
B
BS-A
56
BS-A
56
BS-A
56
PP
60
62
PP
60
PP
60
54
58

FIGURE 3

FIGURE 4

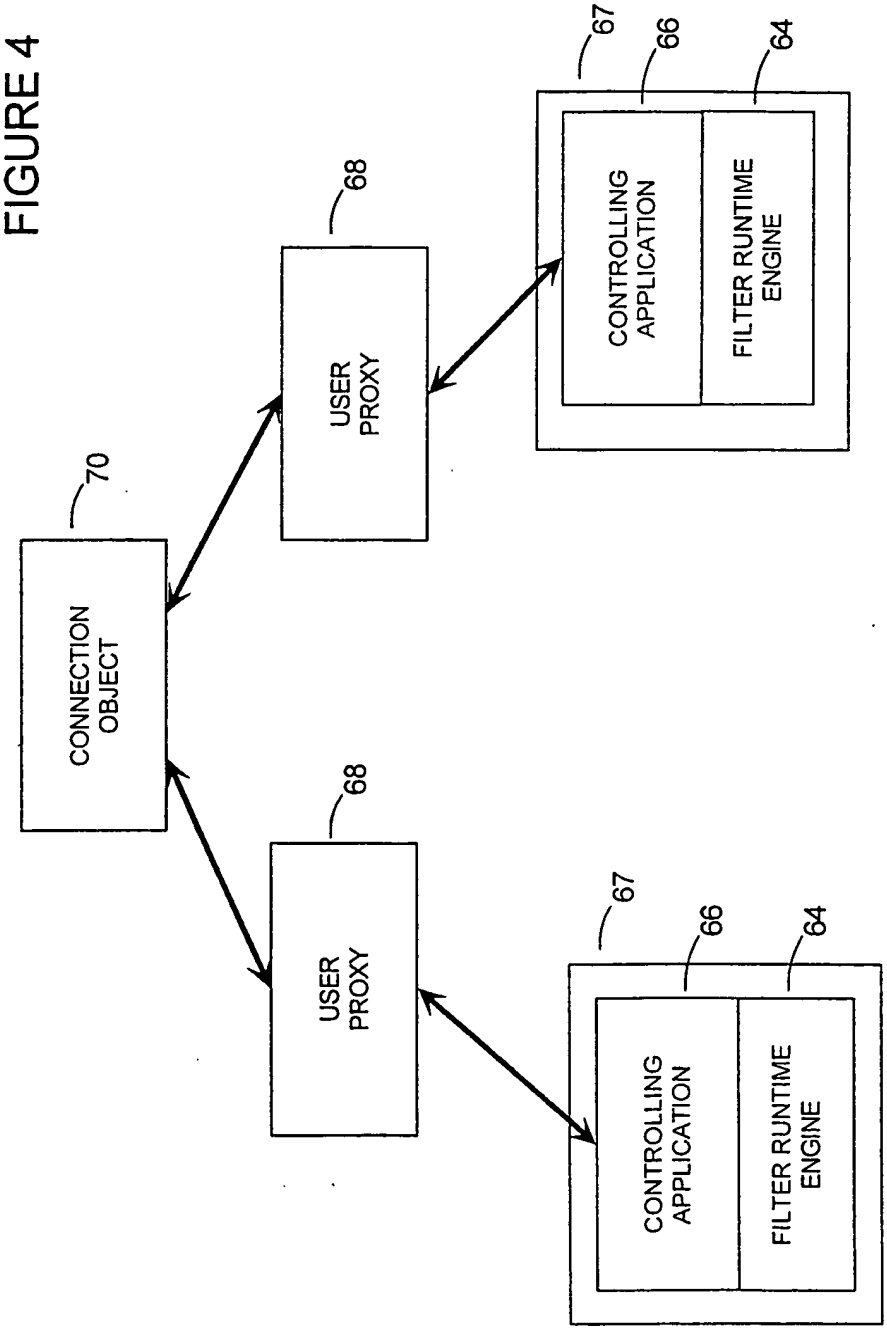
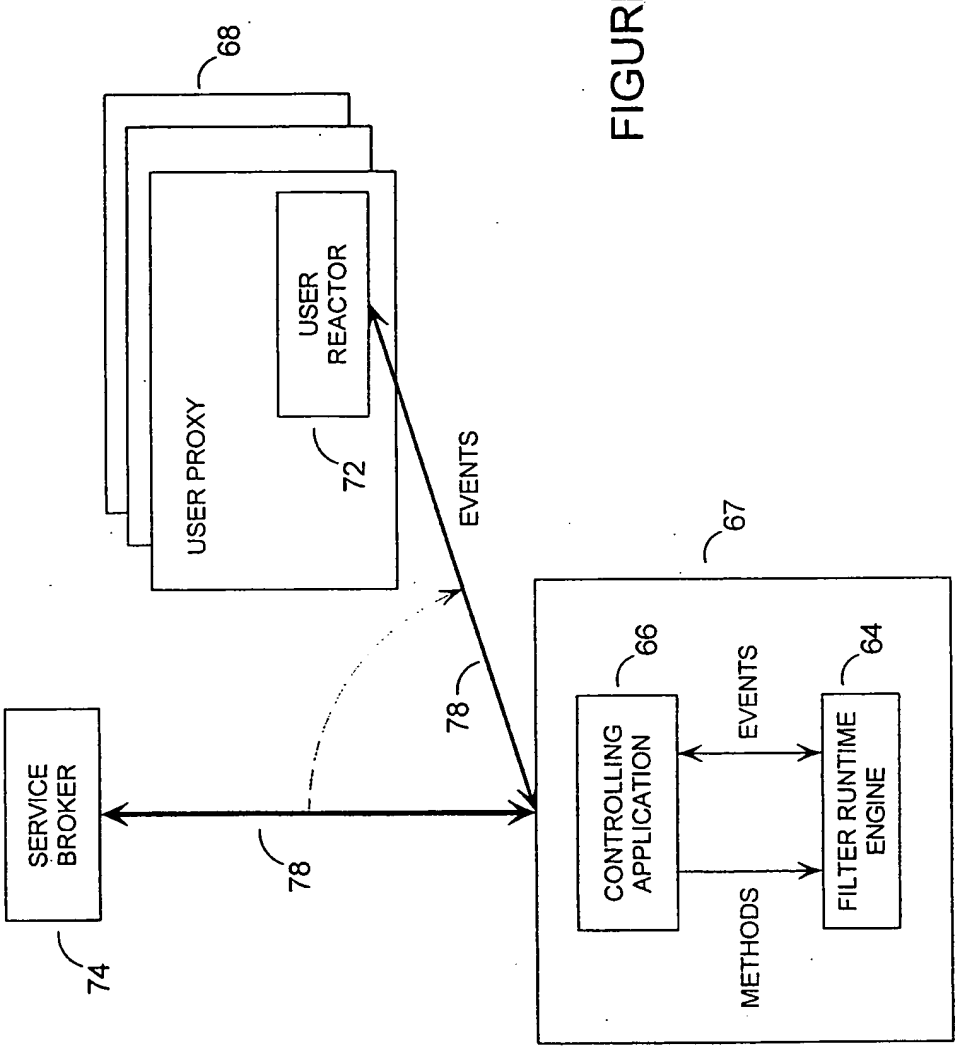

67
66
64
68
CONTROLLING APPLICATION
FILTER RUNTIME ENGINE
USER PROXY
70
CONNECTION OBJECT
68
USER PROXY
67
66
64
CONTROLLING APPLICATION
FILTER RUNTIME ENGINE

FIGURE 5


68
USER PROXY
USER REACTOR
72
EVENTS
67
78
66
CONTROLLING APPLICATION
EVENTS
64
FILTER RUNTIME ENGINE
METHODS
SERVICE BROKER
78
74

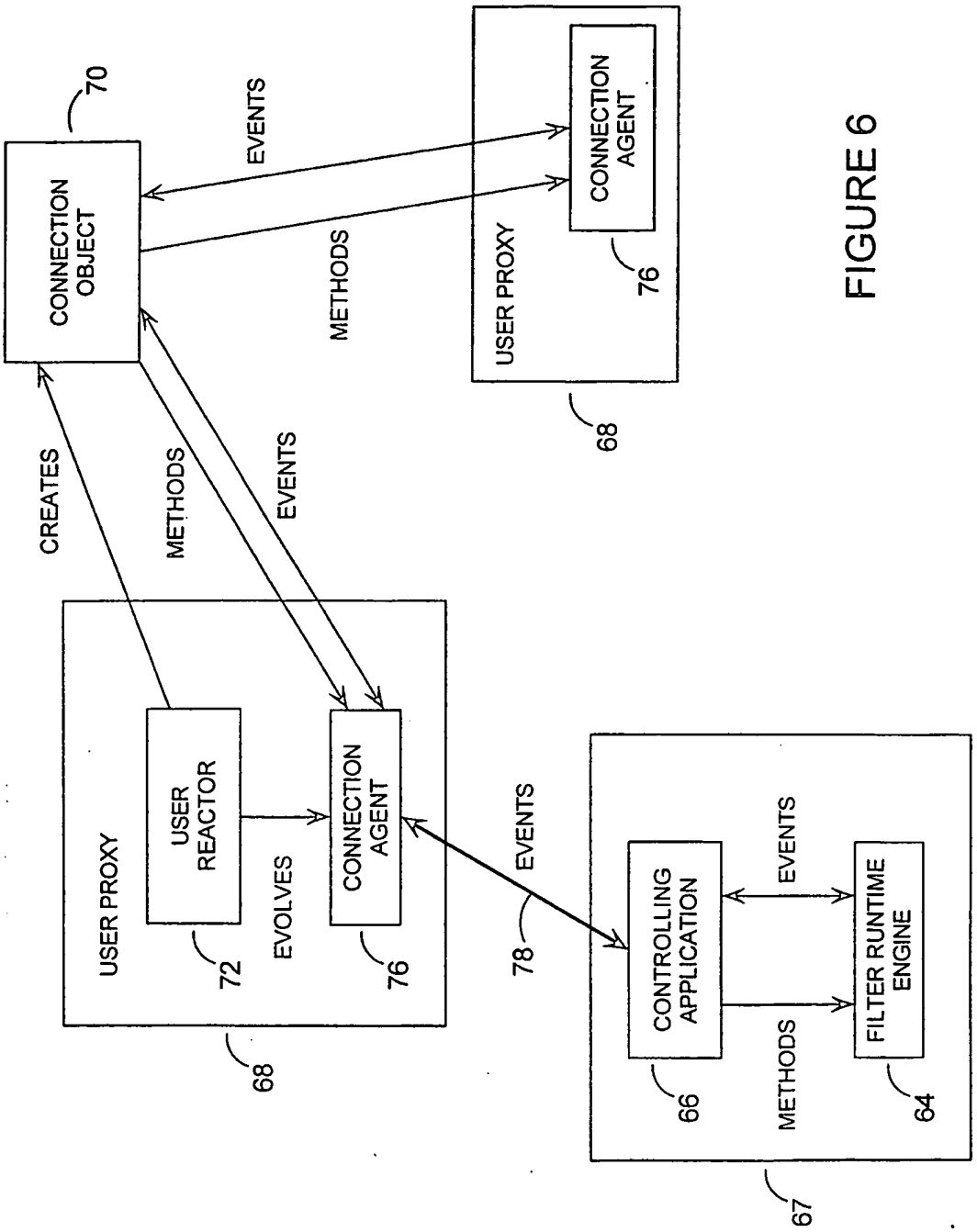

CONNECTION OBJECT
70
EVENTS
METHODS
USER PROXY
CONNECTION AGENT
76
68
CREATES
METHODS
EVENTS
USER PROXY
USER REACTOR
EVOLVES
CONNECTION AGENT
72
76
68
EVENTS
78
CONTROLLING APPLICATION
EVENTS
FILTER RUNTIME ENGINE
METHODS
66
64
67

FIGURE 6

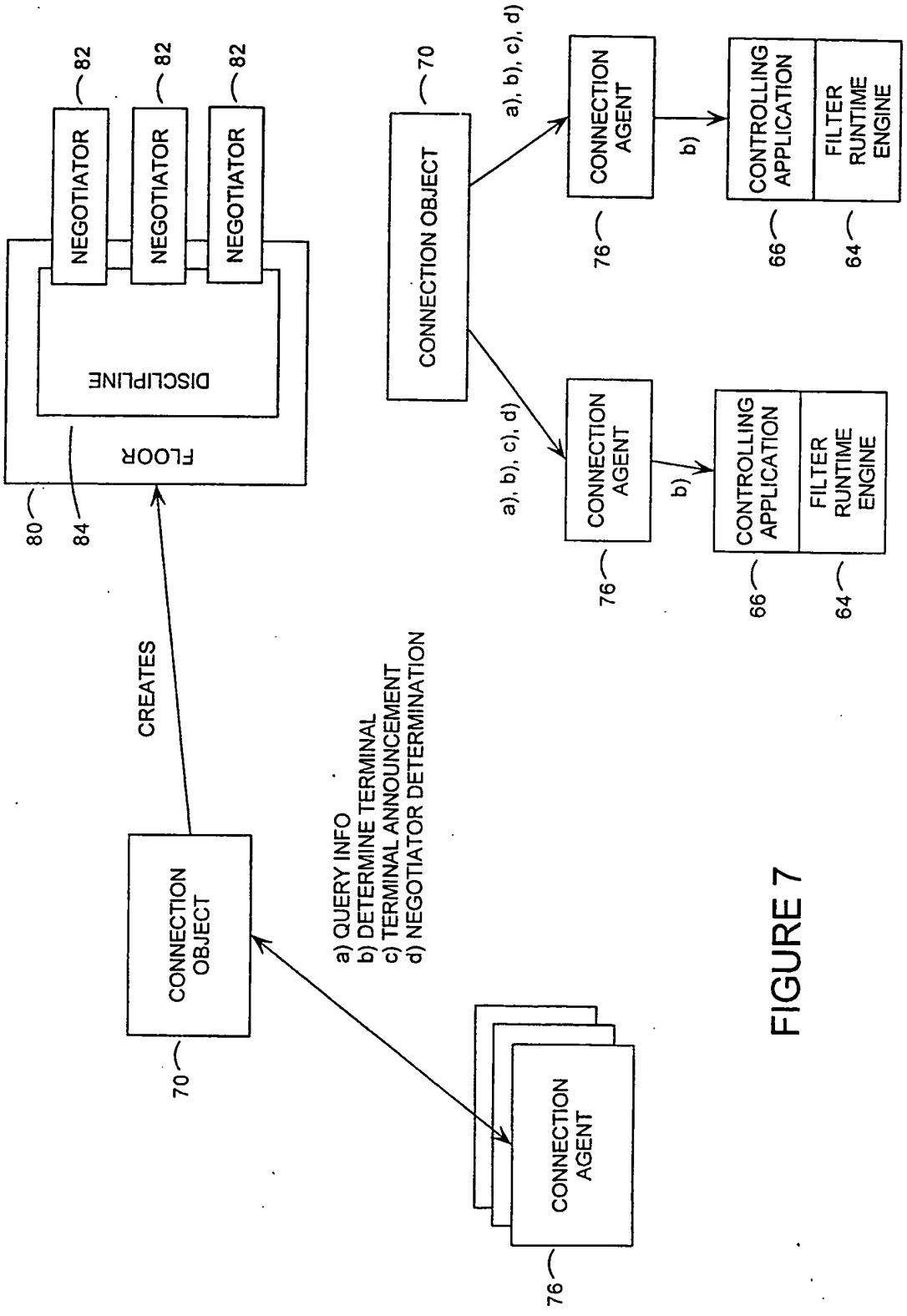

NEGOTIATOR
NEGOTIATOR
NEGOTIATOR
82
82
82
DISCLIPLINE
FLOOR
80
84
CREATES
CONNECTION OBJECT
70
a) QUERY INFO
b) DETERMINE TERMINAL
c) TERMINAL ANNOUNCEMENT
d) NEGOTIATOR DETERMINATION
CONNECTION AGENT
76
CONNECTION OBJECT
70
a), b), c), d)
CONNECTION AGENT
76
b)
CONTROLLING APPLICATION
66
FILTER RUNTIME ENGINE
64
a), b), c), d)
CONNECTION AGENT
76
b)
CONTROLLING APPLICATION
66
FILTER RUNTIME ENGINE
64

FIGURE 7

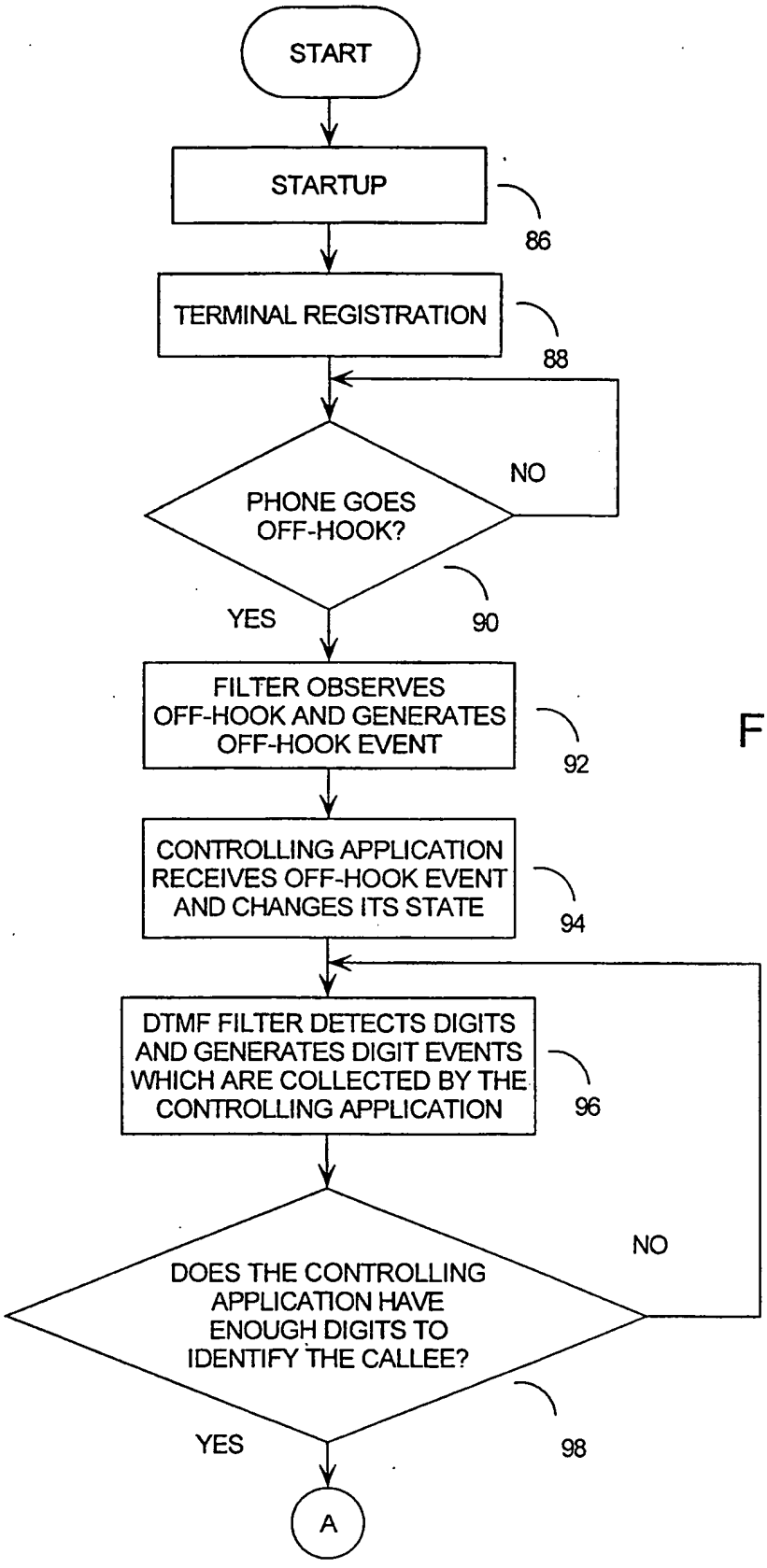

START
STARTUP
86
TERMINAL REGISTRATION
88
PHONE GOES OFF-HOOK?
NO
YES
90
FILTER OBSERVES OFF-HOOK AND GENERATES OFF-HOOK EVENT
92
CONTROLLING APPLICATION RECEIVES OFF-HOOK EVENT AND CHANGES ITS STATE
94
DTMF FILTER DETECTS DIGITS AND GENERATES DIGIT EVENTS WHICH ARE COLLECTED BY THE CONTROLLING APPLICATION
96
DOES THE CONTROLLING APPLICATION HAVE ENOUGH DIGITS TO IDENTIFY THE CALLEE?
NO
YES
98
A

FIGURE 8A

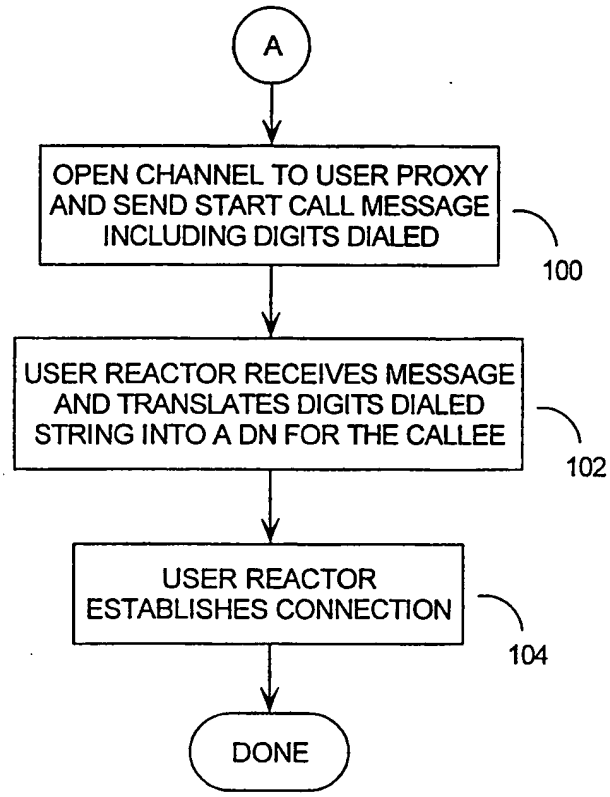

A
OPEN CHANNEL TO USER PROXY AND SEND START CALL MESSAGE INCLUDING DIGITS DIALED
100
USER REACTOR RECEIVES MESSAGE AND TRANSLATES DIGITS DIALED STRING INTO A DN FOR THE CALLEE
102
USER REACTOR ESTABLISHES CONNECTION
104
DONE

FIGURE 8B

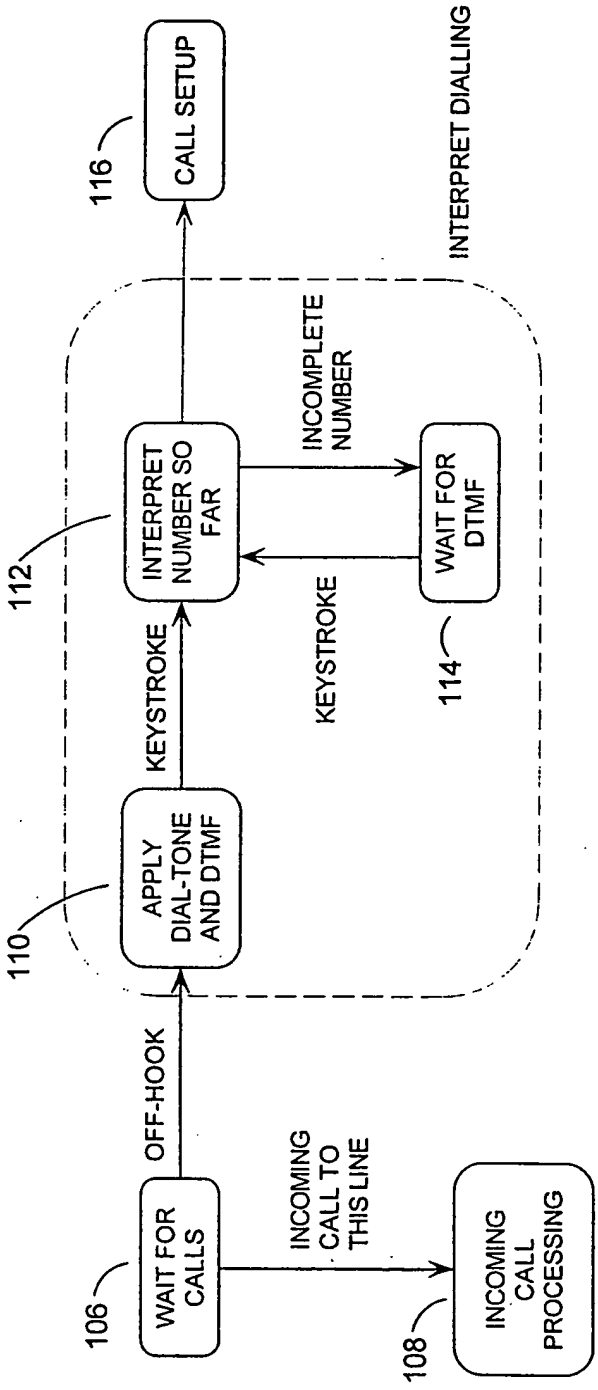

WAIT FOR CALLS
106
OFF-HOOK
INCOMING CALL TO THIS LINE
INCOMING CALL PROCESSING
108
APPLY DIAL-TONE AND DTMF
110
KEYSTROKE
INTERPRET NUMBER SO FAR
112
INCOMPLETE NUMBER
KEYSTROKE
WAIT FOR DTMF
114
CALL SETUP
116
INTERPRET DIALLING

FIGURE 9

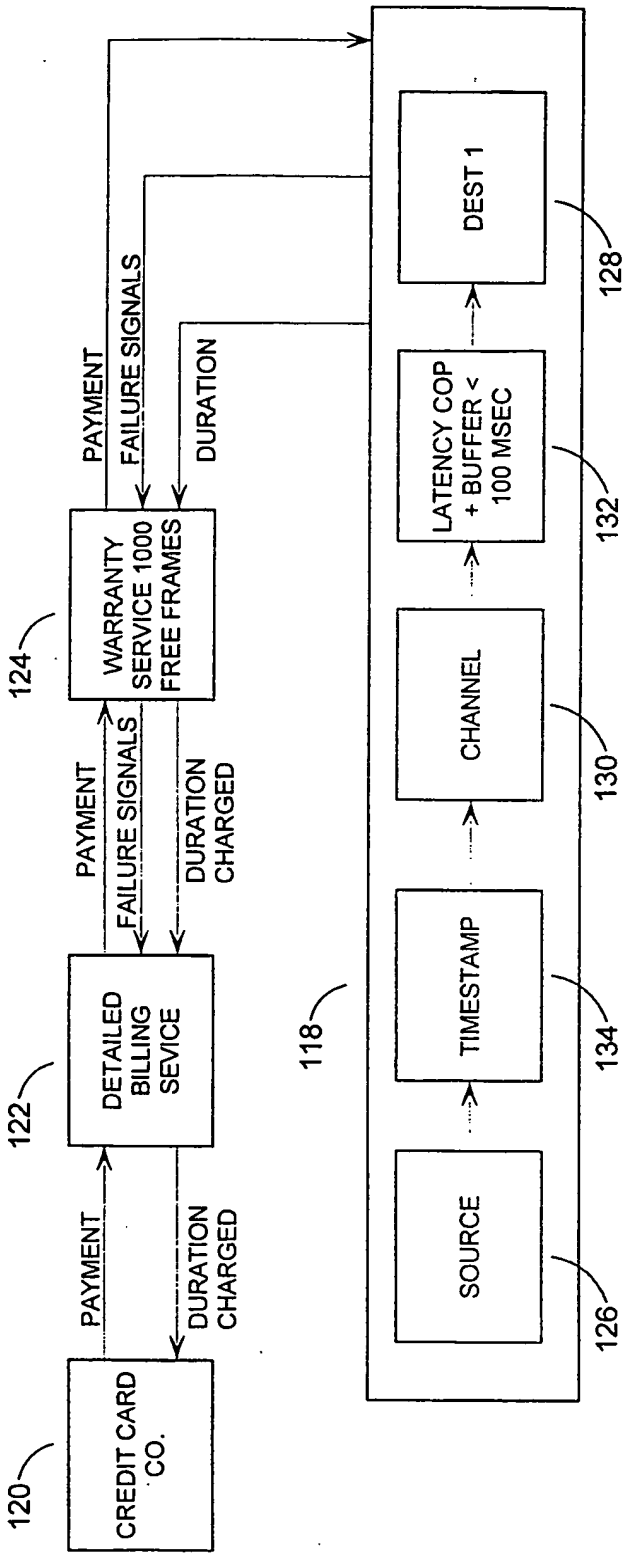

CREDIT CARD CO.
120
PAYMENT
DETAILED BILLING SEVICE
122
PAYMENT
DURATION CHARGED
WARRANTY SERVICE 1000 FREE FRAMES
124
PAYMENT
FAILURE SIGNALS
DURATION CHARGED
FAILURE SIGNALS
DURATION
118
SOURCE
126
TIMESTAMP
134
CHANNEL
130
LATENCY COP + BUFFER < 100 MSEC
132
DEST 1
128

FIGURE 10

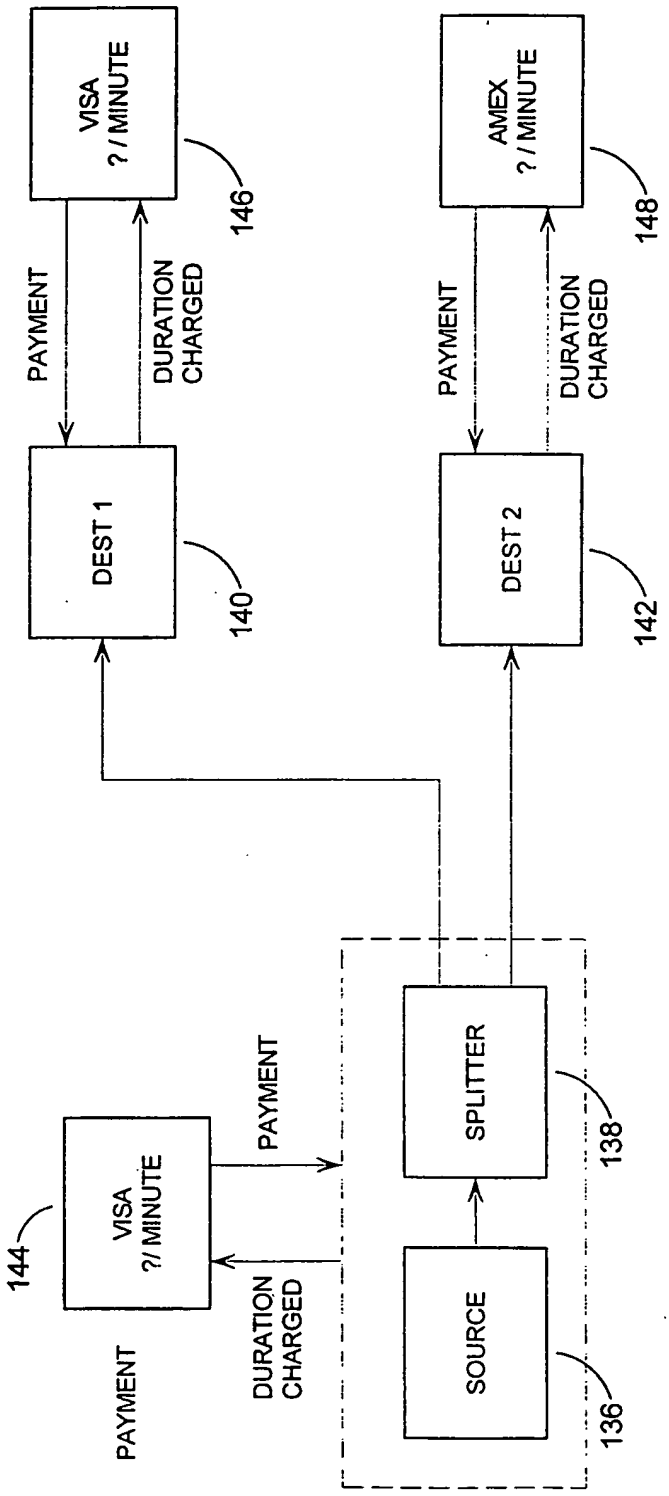

VISA
? / MINUTE
146
AMEX
? / MINUTE
148
PAYMENT
DURATION CHARGED
PAYMENT
DURATION CHARGED
DEST 1
140
DEST 2
142
SPLITTER
138
VISA
?/ MINUTE
144
PAYMENT
DURATION CHARGED
PAYMENT
SOURCE
136

FIGURE 11